Київський національний університет

імені Т.Шевченка

**Звіт**

до лабораторної роботи №2

на тему:

«*Імітаційна модель процесора****»***

***Студента другого курсу***

***Групи К-21***

***Факультету комп’ютерних наук***

***та кібернетики***

***Півня Дениса***

***Київ***

*2019*

**Мета**

Метою даної лабораторної роботи є зрозуміти принцип роботи процесора у спрощеному вигляді, ознайомитися з його архітектурою: способами представлення внутрішніх даних та їх зберігання, представлення цілочисельної інформації в процесорах та пам’яті, програмної моделі, в залежності від адресності процесора, способів зберігання даних, його принципів роботи.

Необхідно розробити програмну модель процесора та реалізувати його імітаційну (тобто комп’ютерну) модель, яка показувала б основні принципи роботи, зчитування команд, виконання обчислень, зберігання даних.

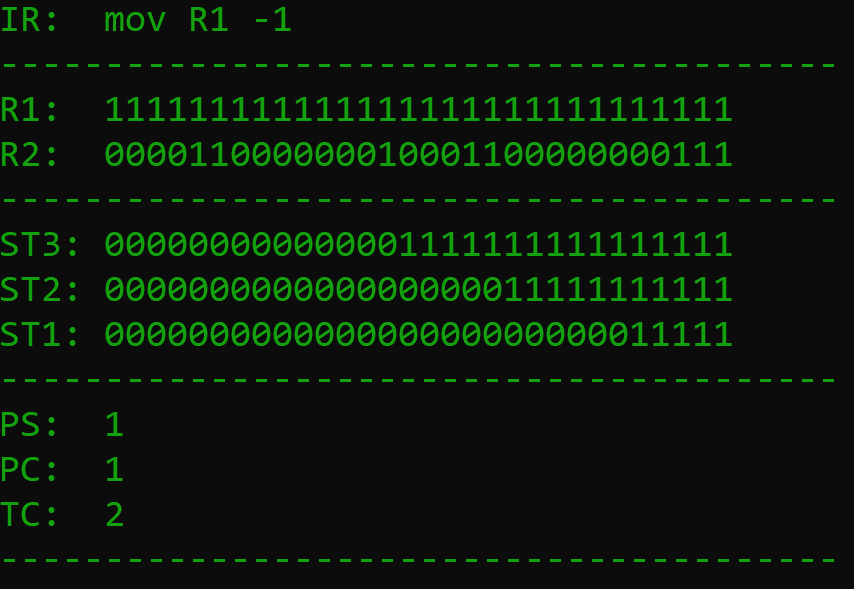
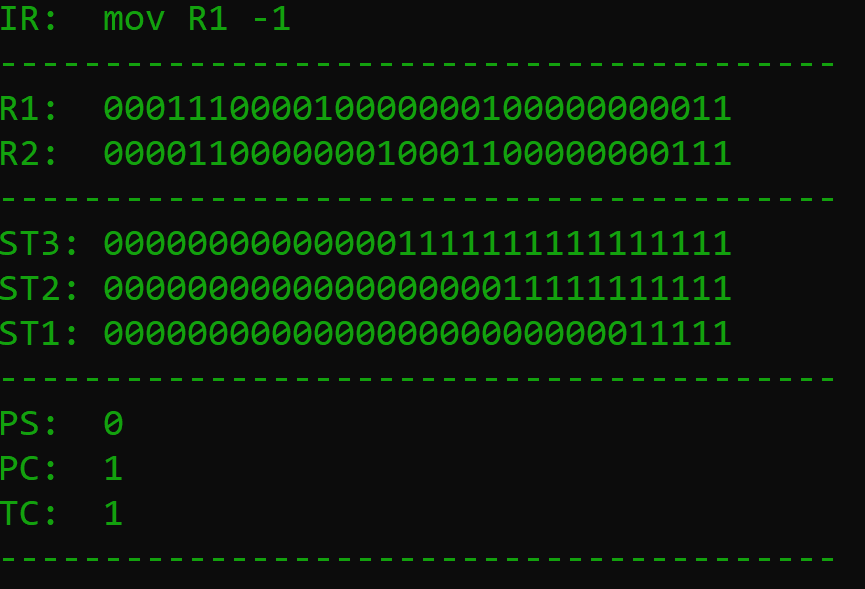
**Основні принципи виконання роботи**

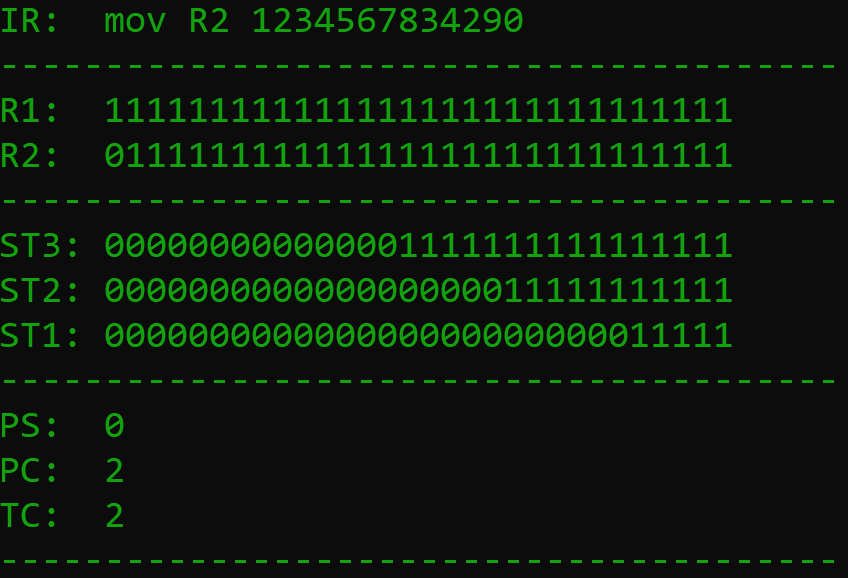
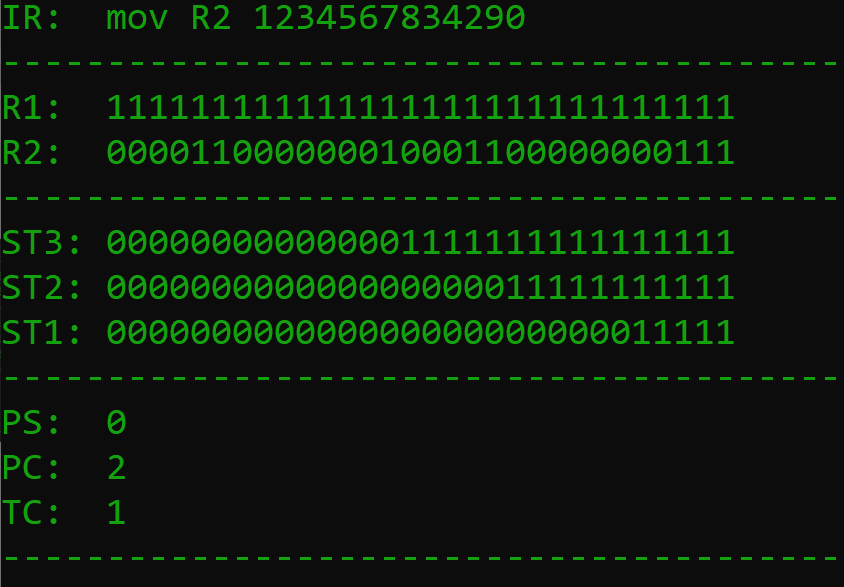
Нижче подані основні принципи написання, не вдаючись до детального розбору коду, який зроблений в розділі «Код програми» у вигляді коментарів до поданого програми:

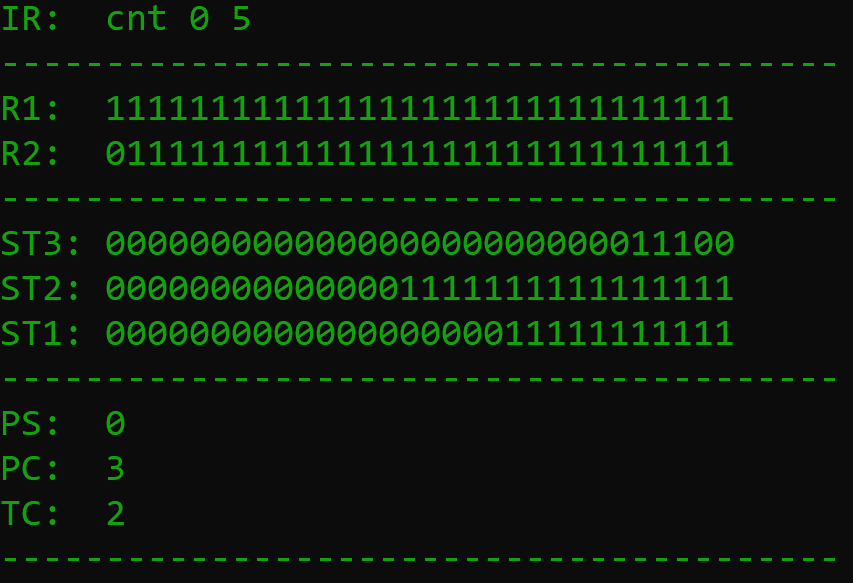
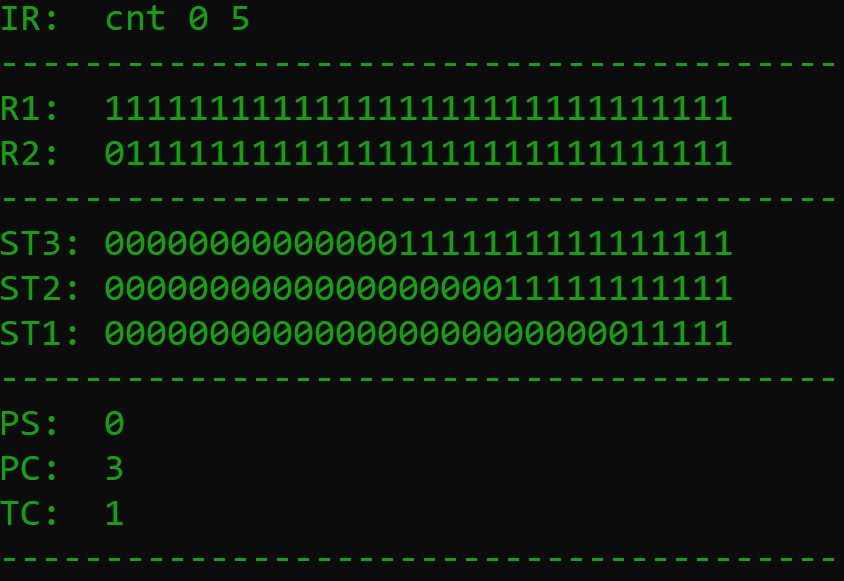
* Виконувався варіант 4.10.13, тобто було створено імітаційну безадресну модель процесора, в якому використовується стек; Результат заноситься у верхівку стека. Можуть бути також декілька регістрів.
* Бітність регістрів та операндів команд – 30 біта;
* Завдання: Кількість одиничних/нульових бітів в залежності від значення *1* чи *0* 2-го операнда команди, представленого у:
* ·         команді безпосередньо чи регістрі для безстекової реалізації;
* ·         верхівці стека в стековій реалізації розміщення операндів.
* Для прикладу, так як і в умовах лабораторної роботи, було взято 2 регістри;
* А також 3 верхні елементи стеку
* Кількість тактів для команд – обрана, відповідно до умов лабораторної роботи, рівною двом: 1-й) занесення поточної команди у регістр команди; 2-й) виконання команди;
* Для підрахунку тактів обрано натискання клавіші Enter.
* Файл інтерпретуємої програми заданий у програмі;
* Кожен регістр на початку роботи програми вже має якесь початкове значення;
* Текстовий файл з командами містить 2 команди: «mov R, значення» та «cnt 0/1 , значення»;
* На екрані для кожного такту представлені такі структурні елементи процесору із даними в них:

1. IR - регістр команди, містить текстовий запис команди, яка є поточною;
2. R1 - перший регістр;
3. R2 - другий регістр;
4. ST3 - вершина стеку (1-й елемент) ;
5. ST2 - 2-й елемент стеку;
6. ST1 - 3-й елемент стеку;
7. PS - регістр статусу, який містить знак останнього результату;
8. PC - регістр лічильника команд;
9. TC - регістр лічильника тактів.

**Лістинг програми**







**Висновок**

Ми ознайомилися з архітектурою та принципами роботи процесора, методами представлення внутрішніх даних та їх зберігання, залежність цих факторів від від адресності процесора, способів зберігання даних, його принципів роботи.

Розробили програмну модель процесора та реалізували його імітаційну (тобто комп’ютерну) модель, яка показувала б основні принципи роботи, зчитування команд, виконання обчислень, зберігання даних, імітували отримання команд за допомогою зчитування їх з текстового файлу, імітували по тактову роботу процесора.

**Додаток. Код програми**

//Piven Denys K-21

#include <iostream>

#include <string>

#include <fstream>

#include <iomanip>

#include <stack>

#include <sstream>

#include <stdlib.h>

#include <locale>

using namespace std;

int bit = 30;

int PC = 1;

void FileReader(fstream& file, string& command, string& operand1, string& operand2);

string tenToBin(string operand2);

string binInvert(string bin);

string binReverse(string bin);

string binAddit(string operand11, string operand22);

string eqalizeLength(string bin, int bit, string operand2);

string eqalizeLengthPositive(string bin, int bit, string operand2);

string count(string operand1, string operand2, int bit);

int main()

{

string source = "file.txt";

fstream file;

file.open(source);

string Cnt = "cnt";

string Mov = "mov";

string R1 = "R1";

string R2 = "R2";

string R1value = "000111000010000000100000000011";

string R2value = "000011000000010001100000000111";

string ST1;

string ST2;

string ST3;

stack <string> ST;

ST.push("000000000000000000000000011111");

ST.push("000000000000000000011111111111");

ST.push("000000000000001111111111111111");

if (!file.is\_open()) cout << "Error, file not open"<<endl;

else

{

while (!file.eof())

{

int TC = 1;

int PS = 0;

string command = "";

string operand1 = "";

string operand2 = "";

FileReader(file, command, operand1, operand2);

if (command == Mov)

{

cout << "IR: " << command << " " << operand1 << " " << operand2 << endl;

cout << "----------------------------------------" << endl;

cout << "R1: " << R1value << endl;

cout << "R2: " << R2value << endl;

cout << "----------------------------------------" << endl;

ST3 = ST.top(); ST.pop();

ST2 = ST.top(); ST.pop();

ST1 = ST.top(); ST.pop();

cout << "ST3: " << ST3 << endl;

cout << "ST2: " << ST2 << endl;

cout << "ST1: " << ST1 << endl;

ST.push(ST1);

ST.push(ST2);

ST.push(ST3);

cout << "----------------------------------------" << endl;

cout << "PS: " << PS << endl;

cout << "PC: " << PC << endl;

cout << "TC: " << TC << endl;

cout << "----------------------------------------" << endl;

cin.get();

if (operand1 == R1) R1value = eqalizeLength(binReverse(tenToBin(operand2)), bit, operand2);

if (operand1 == R2) R2value = eqalizeLength(binReverse(tenToBin(operand2)), bit, operand2);

if (atoi(operand2.c\_str()) < 0) PS = 1;

TC++;

cout << "IR: " << command << " " << operand1 << " " << operand2 << endl;

cout << "----------------------------------------" << endl;

cout << "R1: " << R1value << endl;

cout << "R2: " << R2value << endl;

cout << "----------------------------------------" << endl;

ST3 = ST.top(); ST.pop();

ST2 = ST.top(); ST.pop();

ST1 = ST.top(); ST.pop();

cout << "ST3: " << ST3 << endl;

cout << "ST2: " << ST2 << endl;

cout << "ST1: " << ST1 << endl;

ST.push(ST1);

ST.push(ST2);

ST.push(ST3);

cout << "----------------------------------------" << endl;

cout << "PS: " << PS << endl;

cout << "PC: " << PC << endl;

cout << "TC: " << TC << endl;

cout << "----------------------------------------" << endl;

cin.get();

PC++;

}

if (command == Cnt)

{

cout << "IR: " << command << " " << operand1 << " " << operand2 << endl;

cout << "----------------------------------------" << endl;

cout << "R1: " << R1value << endl;

cout << "R2: " << R2value << endl;

cout << "----------------------------------------" << endl;

ST3 = ST.top(); ST.pop();

ST2 = ST.top(); ST.pop();

ST1 = ST.top(); ST.pop();

cout << "ST3: " << ST3 << endl;

cout << "ST2: " << ST2 << endl;

cout << "ST1: " << ST1 << endl;

ST.push(ST1);

ST.push(ST2);

ST.push(ST3);

ST.push(count(operand1, eqalizeLength(binReverse(tenToBin(operand2)), bit, operand2), bit));

cout << "----------------------------------------" << endl;

cout << "PS: " << PS << endl;

cout << "PC: " << PC << endl;

cout << "TC: " << TC << endl;

cout << "----------------------------------------" << endl;

cin.get();

TC++;

cout << "IR: " << command << " " << operand1 << " " << operand2 << endl;

cout << "----------------------------------------" << endl;

cout << "R1: " << R1value << endl;

cout << "R2: " << R2value << endl;

cout << "----------------------------------------" << endl;

ST3 = ST.top(); ST.pop();

ST2 = ST.top(); ST.pop();

ST1 = ST.top(); ST.pop();

cout << "ST3: " << ST3 << endl;

cout << "ST2: " << ST2 << endl;

cout << "ST1: " << ST1 << endl;

ST.push(ST1);

ST.push(ST2);

ST.push(ST3);

cout << "----------------------------------------" << endl;

cout << "PS: " << PS << endl;

cout << "PC: " << PC << endl;

cout << "TC: " << TC << endl;

cout << "----------------------------------------" << endl;

cin.get();

PC++;

}

}

}

system("pause");

return 0;

}

void FileReader(fstream& file, string& command, string& operand1, string& operand2)

{

string var;

string space = " ";

string line;

getline(file, line);

int i = 0;

try {

while (true)

{

var = line[i]; i++;

if (var == space) break;

command += var;

}

while (true)

{

var = line[i]; i++;

if (var == space) break;

operand1 += var;

}

while (i < line.length())

{

var = line[i]; i++;

operand2 += var;

}

}

catch (...) { cout << "Reading Error"; }

}

string binInvert(string bin)

{

string temp = "";

string zero = "0";

string one = "1";

string bintemp;

for (int i = 0; i < bin.length(); i++)

{

bintemp = bin[i];

if (atoi(bintemp.c\_str()) == 0) temp += one;

if (atoi(bintemp.c\_str()) == 1) temp += zero;

}

return temp;

}

string tenToBin(string operand2)

{

string bin = "";

int Num;

int n = abs(atoi(operand2.c\_str()));

while (n != 0)

{

Num = n % 2;

n /= 2;

ostringstream convert;

convert << Num;

bin += convert.str();

}

if (bin.length() > bit) {

cout << "Error Overfloat" << endl;

bin = bin.substr(0, bit - 1);

}

return bin;

}

string binReverse(string bin)

{

string temp = "";

for (int i = (bin.length() - 1); i >= 0; --i)

{

temp += bin[i];

}

return temp;

}

string binAddit(string operand11, string operand22)

{

string result = "";

string temp;

int mind = 0;

int operand1, operand2;

for (int i = (operand11.length() - 1); i >= 0; i--)

{

temp = operand11[i];

operand1 = atoi(temp.c\_str());

temp = operand22[i];

operand2 = atoi(temp.c\_str());

if (mind == 0)

{

if ((operand1 == 0) and (operand2 == 0)) { result += "0"; mind = 0; }

if ((operand1 == 0) and (operand2 == 1)) { result += "1"; mind = 0; }

if ((operand1 == 1) and (operand2 == 0)) { result += "1"; mind = 0; }

if ((operand1 == 1) and (operand2 == 1)) { result += "0"; mind = 1; }

}

if (mind == 1)

{

if ((operand1 == 0) and (operand2 == 0)) { result += "1"; mind = 0; }

if ((operand1 == 0) and (operand2 == 1)) { result += "0"; mind = 1; }

if ((operand1 == 1) and (operand2 == 0)) { result += "0"; mind = 1; }

if ((operand1 == 1) and (operand2 == 1)) { result += "1"; mind = 1; }

}

}

return binReverse(result);

}

string eqalizeLength(string bin, int bit, string operand2)

{

string delta = "";

string temp = "";

if (atoi(operand2.c\_str()) >= 0) return (eqalizeLengthPositive(bin, bit, operand2));

for (int i = 0; i < (bit - bin.length()); i++) temp += "0";

bin = temp + bin;

for (int i = 0; i < (bit - 1); i++) delta += "0";

delta += "1";

return(binAddit(binInvert(bin), delta));

}

string eqalizeLengthPositive(string bin, int bit, string operand2)

{

string temp = "";

if (atoi(operand2.c\_str()) < 0) temp = "1";

else temp = "0";

for (int i = 0; i < (bit - bin.length() - 1); i++) temp += "0";

bin = temp + bin;

return(bin);

}

string count(string operand1, string operand2, int bit)

{

int count = 0;

string temp;

int temp2;

if (atoi(operand1.c\_str()) == 0)

{

for (int i = 0; i < operand2.length(); i++) {

temp = operand2[i];

temp2 = atoi(temp.c\_str());

if (temp2 == 0) count++;

}

}

if (atoi(operand1.c\_str()) == 1)

{

for (int i = 0; i < operand2.length(); i++) {

temp = operand2[i];

temp2 = atoi(temp.c\_str());

if (temp2 == 1) count++;

}

}

string count2;

int Number = count;

ostringstream convert;

convert << Number;

count2 = convert.str();

return (eqalizeLength(binReverse(tenToBin(count2)), bit, count2));

}